

# Jobs Scheduling In Grid Computing Network, With Hybrid Particle Swarm Optimization Algorithm

## 1. Abbas Akkasi

Computer Engineering Department of Islamic  
Azad University Science and Research Branch,  
Tehran, Iran

Abbas.Akkasi@gmail.com

## 2. Sayed Ali Talebnia Jahromi

Islamic Azad University  
Larestan Branch, Iran

A.talebnia@yahoo.com

## 3. Ahmad Mosallanjad

PhD Department of Computer Engineering  
Faculty of Engineering Sepidan Branch, Islamic Azad  
University

Sepidan, Iran

mosala@iausepidan.ac.ir

## 4. Sayed Ali Rahimi

Islamic Azad University  
Arak Branch, Iran

S\_A\_Rahimi@yahoo.com

## 5. Kalil Moayedfar

Islamic Azad University  
Larestan Branch

Larestan, Iran

hamedmyd@gmail.com

**Abstract—** Grid Network can be considered as a computational framework for addressing the growing computing requirements. In This paper, a new particle swarm optimization based approach for time scheduling in grid computing is represented. Falling into local optima and the low speed of convergence are two main issues which standard PSO<sup>1</sup> Algorithm has been faced to. Using hybrid algorithm with utilizing fuzzy logic can be regarded as an alternative approach to dissolve such problems. The goal of this approach is to optimal and dynamically scheduling of job completion in as minimum time as possible and furthermore utilizing resource in an effective manner.

**Keywords—** Particle Swarm Optimization, Fuzzy System, Grid computing , hybrid particle swarm optimization

## I. INTRODUCTION

A computing network is some large scale of heterogeneous set of geographical distributed independent systems which are connected together by the means of high bandwidth and low latency network [1]. Job sharing can be considered as the main application of grid computing networks. There are a variety of dynamic resources In a grid computing network , which in any time can be added or removed from the network, and such changes can frequently occur with the passing of time. For this reason, the importance of performance and complexity of such issues attracted the researchers' attention. The recent results have shown that optimally assigning a job to a computing network and in a practical scheduling manner can lead to lowering the job time completion [2-4]. In this paper, we will introduce a new particle swarm optimization based (PSO) approach [5] for job scheduling in grid computing network.

<sup>1</sup> Particle Swarm Optimization

This paper is organized as follows. In section 2, job-scheduling formulation is presented. Section 3, dynamic scheduling tasks in PSO-based grid networks provided, and in section 4, the suggested algorithm is presented and in section 5, the evaluation results of algorithm is presented and finally in section 6, there is the conclusion.

## II. DYNAMIC JOB SCHEDULING IN PSO-BASED GRID NETWORK

Hybrid optimization issues have great importance in practical applications and recently, many researches with progresses in inspiration of nature and multi-agent systems for scheduling problems have been done in this regard, and consequently this had led to a significant increase of search space amount and has necessitated the need to real-time scheduling and motivational researches for addressing scheduling problem through using nature inspired technology. In this section, we develop a fuzzy based discrete particle swarm optimization for addressing job-scheduling problem.

### A. Standard particle swarm optimization

Particle swarm algorithm is inspired by social pattern of organism and the interaction inside big Group of birds, fishes and bees swarms and even social behavior of human for example can be regarded as such behaviors. Such an algorithm can be easily evolved and used for different functions of optimization problems. The standard model of PSO is comprised of particle swarm, which is initialized with random candidate solutions. Such models for searching a new solution, frequently moves throughout the problem domain. A position is assigned to each particle shown with a position vector and velocity of each particle in a similar manner is shown by velocity vector. Each particle stores its best current position in a vector. The best vector position which has ever occurred among the swarm is stored in another vector. According to the previously acquired experiences from the best position in the past, a particle decides whether to

move. In particle swarm model, the particles search the solutions in problem domain.

*B. A hybrid fuzzy method with PSO*

According to the previously mentioned discussions in above and the familiarities we have obtained from PSO problems, we can say that falling into local optima and the low speed of convergence are the main problems in this regard. In order to solve the first problem, we should deceive a solution in such a way that local optima could be detected and prevented [6]. Several approaches are proposed to succeed in dealing with such problems. In [7] a function is used to make an uncertainty in particle velocity, in order to prevent falling into local optima, provided that the particle velocity is below a certain threshold but its fitness value be acceptable. In [8,9] and a fuzzy non-linear function is used to change inertia coefficient and in the event that the obtained fitness value is not acceptable and inertia coefficient is reduced too, this function causes the coefficient to be increased and with the increment of particle velocity, the possibility of universal search is increased too. In [8, 10] preventing from falling into local optimization as much a possible is reached by identification of two adjacent local peak. All the mentioned approaches somehow will result in preventing from falling into local optimal. In this paper we represent a new approach which in fact is a non-linear function for inertia coefficient ( $w$ ) and is another function for velocity coefficients ( $C1,C2$ ). In what follows, we will shortly describe our proposed hybrid approach.

*C. creating adaptive fuzzy confusion in PSO*

In [7], in order to prevent from falling into local optimization, a new method namely TPSO<sup>2</sup> is presented. Provided that the particle velocity is less than  $V_c$  threshold, a new velocity is assigned to particle with Eq. (1). The current velocity of particle is calculated with the aim of relation (2).

$$v_{id}(t + 1) = w \cdot v_{id} + c_1 \cdot \text{rand}_1(pbest_{id}(t) - x_{id}(t)) + c_2 \cdot \text{rand}_2(gbest_{id}(t) - x_{id}(t)) \quad (1)$$

$$\hat{v} = \begin{cases} v_{id} & \text{if } |v_{id}| \geq v_c \\ u(-1,1)v_{max}/p & \text{if } |v_{id}| < v_c \end{cases} \quad (2)$$

In relation (2),  $u(-1,1)$  is a random number with uniform distribution within  $[-1,1]$  and  $P$  is measurement parameter<sup>3</sup> of two particle fluctuation range in accordance to  $V_{max}$ . The location change of each particle is depend upon  $V_c$  and  $P$  parameters. If  $V_c$  be small, the particle fluctuation is lowered in this case and the possibility of falling into local optimization will be increased. But the magnitude of  $V_c$  allows the particle to jump from each position to another position. Thus, great value of  $V_c$  makes universal searching possible and small amount of that, increases local searching possibility. Also,  $P$  directly changes particle fluctuation. big value of  $P$  will cause the particle velocity to be decreased and allows the particle not to jump to the big local optimal which exist inside search

space and the possibility of finding a solution is more in there. However, if  $P$  value is small, in this case the path of particle movement will be pruned. In fact, this parameter adjusts the amount of local and universal search with increasing and decreasing its value. In this algorithm, we consider to variable CBPE (3 is the best current performance evaluation rate)<sup>4</sup> and current velocity<sup>5</sup> of 4 particles as a fuzzy system's input. CBPE illustrates the fitness value of particle in current state and CBPE<sub>min</sub> illustrates the best fitness value which the particle has ever obtained and CBPE<sub>max</sub> illustrates the worst fitness value the particle has ever obtained. In Eq. (3), NCBPE is a normalized value of best, worst and current value of fitness function that takes values within  $[0-1]$ .

$$NCBPE = \frac{CBPE - CBPE_{min}}{CBPE_{max} - CBPE_{min}} \quad (3)$$

There are two output parameters for this system in which, the former is  $P$  and the latter is  $V_{ck}$  (the parameter which controls the fluctuation rate of particle) which with the aim of relation (4) updates the rate of threshold velocity.

$$v_c = e - [10(1 + V_{ck})] \quad (4)$$

There are two widely used procedures for inference. The first type is Mamdani fuzzy interface system that was represented by Mr. Ebrahim Mamdani in 1975, and the second type is Takkagi sugeno which was presented in 1985. This two method in most aspects are similar, like input fuzzy fication and fuzzy operators, but the main difference in between them is that the output of sugeno method is a member of the functions, which can be constant or linear, but in Mamdani Inference we expect the output to be of membership function of fuzzy sets [11]. In designing our system and in accordance to the expected results, we have used Mamdani's approach.

Our Fuzzy inference system is implemented in accordance to the method of Mamdani. This system has two inputs and two outputs which are illustrated in Fig. (1).

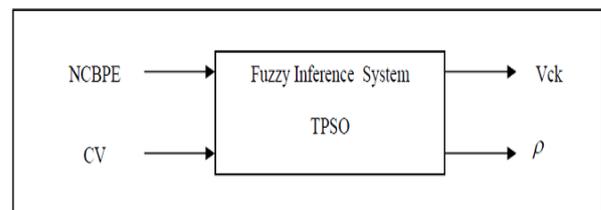


Fig. 1. TPSO Fuzzy inference system [13].

*D. A non-linear function for middle weight W*

According to the previous sections, one of the problems in standard PSO was linear convergence of algorithm, which causes the convergence speed to be reduced gradually. To address this problem, we can use non-linear functions for middleweight. As long as there are several parameters for decision-making, thus

<sup>2</sup> Turbulent Particle Swarm Optimization – TPSO

<sup>3</sup> scaling factor

<sup>4</sup> Current Best Performance Evaluation - CBPE

<sup>5</sup> Current Velocity - CV

fuzzy functions can have a special role in this regard. Such functions were parented in [12] for the first time. In an approach we call WPSO, the essence of decision-making is different from other approaches. According to the NCBPE parameter definition, it can be said that the convergence rate is determined by the means of this parameter and if such parameter have a value close to zero, it can be said that the convergence result would be favorable. Thus in this algorithm, the basis of decision-making is in accordance to  $d1$  and NCBPE Parameters. In Fig. (2), a schematic representation of such system is represented. In this approach and in accordance to  $d1$  and NCBPE, a new value is determined for  $W$ .

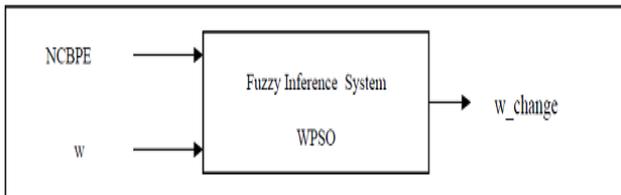


Fig. 2. WPSO fuzzy inference system [13].

### III. THE ALGORITHM

#### A. FPSO<sup>6</sup> Hybrid approach

In the algorithm we call FPSO1, inertia coefficient is a fuzzy function which take three parameters namely  $d1$ ,  $d2$  and NCBPE as input and  $W$  is the output of function which is illustrated in Fig. (3).

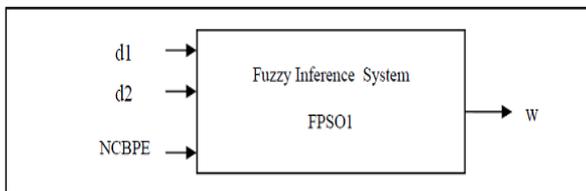


Fig. 3. fuzzy function of FPSO1 [13].

$$d1 = |p_{best} - x| \quad d2 = |g_{best} - x| \quad (5)$$

In relations to (5), two parameter  $d1$  and  $d2$  represent the particle closeness to the best local and universal experience which can recite closeness level of local and universal level and NCBPE which was described in Section C. To introduce such parameters, we have used low, moderate and high lingual tags in which, the range of  $d1$  and  $d2$  them are determined in accordance to the size of search space. Thus, this parameters are based on fuzzy system decision making to determine value with in  $[0 \ 1]$ . Nevertheless, it must be noted that the selection of fuzzy rules have a direct effect on obtained results. Table (1) illustrates some of rules used in this system.

TABLE 1. Some of fuzzy rules in FPSO1 algorithm [13].

Rules	Input			Output
	d1	d2	NCBPE	W
1	High	High	High	High
2	Low	Low	Low	Low
3	Low	Low	Not low	High
4	Low	Low	Medium	Medium

There can be considered a variety of rules for this function, but the experiment has shown that the abundance of rules not only does not have a major effect on the results, but also can consider them as a hotspot for the quality of the selected rules which will produce the most related results. The rules in table 1 can be stated as follow:

Rule 1: in this rule, because of the high value of  $d1$ ,  $d2$  and NCBPE, its velocity must be increased, thus a bigger value of  $w$  is chosen to rapidly reach to the results.

Rule 2: if  $d1$ ,  $d2$  and NCBPE are small. It means that the particle is close to the best local and universal experience and from the other side, the fitness value of that is in an acceptable range, which in this case, a low value is chosen for  $W$ , in such a way that the particle will search around this position in order to reach to the best result.

Rule 3: if  $d1$  and  $d2$  are low, but NCBPE is not (it means it can be average or high), it means that the particle is close to the best local and universal position but does not have an appropriate distance with universal optimal and in this case, the particle is not in a desirable condition and hence an average value is closed for  $W$  [13].

As a result, in GRID network, the job is done in accordance to the priority and vicinity to optimal solution.

#### B. Algorithms evaluation

In the algorithm, the basis is on changing the current particle's status. The algorithm, standard PSO algorithm and WPSO and TPSO<sup>7</sup> Algorithms are simulated with MATLAB 7.1 and have been evaluated with benchmark Functions. This functions are the most widely used function which have been used in similar evaluations[7,8,14,15,16,17]. In table (2) the used benchmark functions are illustrated. In addition, in table (2), the diagrams related to each benchmark function is illustrated.

In this evaluation, 20 particle with 5, 10, 15 dimension and  $c1=c2=2$  are considered. For TPSO and SPSO, the range  $[0.9, 0.1]$  is considered for  $W$  and after 4000 time iterations, the desired results we obtained. The above algorithms were run 10 times independently and the presented results are obtained in 10 iterations [18]. In tables (3) to (8), the results of such simulations for benchmark functions and their different aspects is represented. The best and the middle solution obtained for the proposed algorithm

<sup>6</sup> Fuzzy PSO

<sup>7</sup> Turbulent Particle Swarm Optimization

and standard PSO algorithm and existing algorithms are compared. All benchmark functions are minimized.

TABLE 2. Standard benchmark function [13].

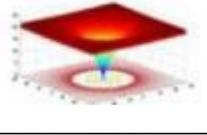
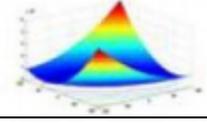
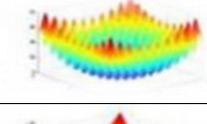
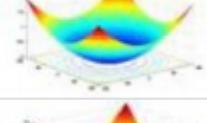
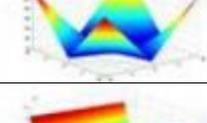
Function name	Formula	Domain	Diagram(n=2)
<b>Ackley</b>	$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)}$	$[-32.32]^n$	
<b>Quadric</b>	$f(x) = \sum_{i=1}^n \left( \sum_{j=1}^i x_j^2 \right)$	$[-100,100]^n$	
<b>Rastrigin</b>	$f(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	$[-5.12,5.12]^n$	
<b>Sphere</b>	$f(x) = \sum_{i=1}^n x_i^2$	$[-10,10]^n$	
<b>Schewefel's problem 2.22</b>	$f(x) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	$[-10,10]^n$	
<b>Rosenbrock</b>	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^n$	

TABLE 3. The results obtained for different dimensions for Ackley function [13].

Algorithm	dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	8.88e-16	0.056	2.66e-15	0.076	2.15e-12	0.219
<b>TPSO</b>	4.26e-8	0.019	1.25e-5	0.044	4.25e-2	1.198
<b>WPSO</b>	1.64	3.349	1.96	4.011	2.43	4890
<b>SPSO</b>	2.31	5.488	3.05	6.457	4.72	7.664

TABLE 4. The results obtained for different dimensions for quadric function [13].

Algorithm	dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	4.88e-149	88.03	2.83e-101	73.20	4.46e-46	47.93
<b>TPSO</b>	3.49e-15	11.73	4.34e-10	16.97	3.56e-8	35.13
<b>WPSO</b>	415.29	89.42	7.51	63.58	6.35e-1	49.67
<b>SPSO</b>	2.47e-15	9.31	1.13e-13	18.18	2.55e-8	33.73

TABLE 5. The results obtained for different dimensions for Rastrigin function [13].

Algorithm	5dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	0.301	0.777	0.388	2.328	1.203	2.031
<b>TPSO</b>	1.479	3.710	1.683	3.912	1.923	6.560
<b>WPSO</b>	1.549	1.773	2.063	2.219	2.845	3.599
<b>SPSO</b>	6.324	14.586	6.532	15.012	6.991	15.558

TABLE 6. The results obtained for different dimensions for shpere function [13].

Algorithm	5dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	9.18e-275	47.55	3.66e-223	50.51	2.11e-84	120.65
<b>TPSO</b>	2.07e-116	261.12	6.81e-60	280.19	7.70e-41	300.86
<b>WPSO</b>	8.86e+2	183.49	8.55e+2	125.71	6.26e+2	73.06
<b>SPSO</b>	4.16e-133	262.86	1.35e-45	287.23	2.43e-26	295.48

TABLE 7. The results obtained for different dimensions for schwefel's problem 2.22 function [13].

Algorithm	5dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	1.46e-159	1.02	2.45e-123	7.37	6.12e-99	128.25
<b>TPSO</b>	1.88e-67	3.85	4.41e-34	40.95	1.81e-31	667.03
<b>WPSO</b>	2.79e-3	0.23	0.313	8.68	0.854	32.13
<b>SPSO</b>	5.26-e10	6.01	6.61e-2	60.66	0.402	8194.24

TABLE 8. THE RESULTS OBTAINED FOR DIFFERENT DIMENSIONS FOR ROSENBROCK FUNCTION [13].

Algorithm	5dimensions		10 dimensions		15 dimensions	
	Best value	Middle value	Best value	Middle value	Best value	Middle value
<b>FPSO1</b>	0.0278	20677	0.3214	24551	0.7943	62924
<b>TPSO</b>	0.1628	451038	1.3962	525734	2.6661	657842
<b>WPSO</b>	4.8573	6919	7.6965	7124	9.3214	7854
<b>SPSO</b>	33.4833	38405	51.2411	99851	125.1253	175491

The results obtained from tables (3) to (8) show that the behavior of the algorithm FPSO1 in all used benchmark functions is optimized and the related diagram to this algorithm is lower than TPSO, WPSO and SPSO. The tangible difference exists in information table, Shows the superiority of the algorithm (FPSO1). This algorithm has been successful in detecting local optimal and was able to release itself from local optimal trap. Generally it can be concluded that the parameters which the algorithm uses to fuzzy decision making (d1 and d2) in a finer manner can detect local optimal and rapidly converge to optimal solution.

#### IV. PROBLEM FORMULATION

The resource within a computing network is variable and dynamic and it is required to be aware of the available resource at any time. Chung and chang [19] introduced a prototype of grid resource information monitoring (GRIM) to manage grid resource in a large

scale to dynamic access to them and management of them. In a grid environment, usually we can easily get information about nodes speed in grid network, but determining the required time to fulfill a job is completely complex. To dissolve this problem in an algorithm, we need to dynamically estimate the length of job from user program or the past information.

To clarify, some key words are defined as below:

Grid network node (computing unit): a network node is a collection of computing resources with limited capacity. This may be a simple machine, a workstation, a super computer or asset if work stations. Computing capacity of node is depend on the number of processors, the amount of memory, the main storage drive and other specifications, which the processing power of each node is expressed as cycles per unit of time (CPUT).

Jobs and operations:

A job is considered as a unit set of atomic tasks/operations. The operations have one processing length which is expressed as a number of cycles.

Scheduling and scheduling problem:

A scheduling is a mapping of jobs in specific time intervals of each grid node, which is specified with a series of machines, a set of operations, optimality criteria, environmental specifications and other limitations.

We have considered independent jobs as  $J_j (j \in \{1,2,3, \dots\})$  and the node in heterogenous grid network nodes as  $G_i (i \in \{1,2,3, \dots\})$ . Each job  $J_j$  needs to be processed and node  $G_i$  has a calculation speed. Each job  $J_j$  in a Grid Network  $G_i$  is processed until it is completed. As all the nodes in each stage are equal, to define scheduling if is sufficient to determinate the completion time of all jobs. To formulate our goal, we have defined  $c_{i,j} (i \in \{1,2,3, \dots, m\}, j \in \{1,2,3, \dots, n\})$  as job finish times, in means when  $G_i$  node finishes job  $J_j$ . The simplest rule to reduce execution time is to assign the longest job to the fastest node.

#### V. IMPLEMENTING PROPOSED ALGORITHM FPSO1 IN GRID NETWORK

In this section, we will design a schema based on discrete FPSO1 to address job scheduling problem in grid network. In this method, the vectors are generalized to fuzzy matrices in which, such matrices are used to represent the position and the velocity of particle for job scheduling. Let  $J = \{j_1, j_2, \dots, j_n\}$  and  $G = \{G_1, G_2, \dots, G_m\}$  is relational fuzzy scheduling from  $G$  to  $J$  which can be expressed as follow:

$$s = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \dots & s_{mn} \end{bmatrix}$$

In which,  $S_{ij}$  is membership degree of its element  $G_i$  in  $G$  domain and  $j$  th element  $J_j$  in  $J$  domain in  $S$  relation. For successful application of PSO, one key issue is finding a roadmap for the existing problem in PSO particles, which directly affects the applicability and performance. We suppose job and networks node are sorted in an ascending order and in term of job length and the nodes processing speed. The information related to possible job lengths may be comprised of data logs, user defined strategy or loading profile.

$$x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$$

So, the element of matrix  $X$ , should follow the following (6), (7) conditions:

$$x_{ij} \in [0,1], i \in \{1,2,3, \dots, m\}, j \in \{1,2,3, \dots, n\} \quad (6)$$

$$\sum_{i=1}^m x_{ij} = 1, i \in \{1,2,3, \dots, m\}, j \in \{1,2,3, \dots, n\} \quad (7)$$

Also, the matrix of particle velocity vector is defined as below:

$$v = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{bmatrix}$$

The position matrix may violate the conditions after some iteration. Thus it is necessary for positions matrices to be normalized. We should convert all negative elements to zero. If all elements inside a matrix column are equal to zero, it is required to determine them with a series of random numbers within  $[0, 1]$ . Therefore, the resulted matrix will be as below, which does not violate any condition.

$$X_{normal} = \begin{bmatrix} \frac{x_{11}}{\sum_{i=1}^m x_{i1}} & \frac{x_{12}}{\sum_{i=1}^m x_{i2}} & \dots & \frac{x_{1n}}{\sum_{i=1}^m x_{in}} \\ \frac{x_{21}}{\sum_{i=1}^m x_{i1}} & \frac{x_{22}}{\sum_{i=1}^m x_{i2}} & \dots & \frac{x_{2n}}{\sum_{i=1}^m x_{in}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{m1}}{\sum_{i=1}^m x_{i1}} & \frac{x_{m2}}{\sum_{i=1}^m x_{i2}} & \dots & \frac{x_{mn}}{\sum_{i=1}^m x_{in}} \end{bmatrix}$$

In this way, the position matrix shows a potential solution for scheduling; we should decode a fuzzy matrix and get an applicable solution.

We use sign array to register the selected column from matrix and use scheduling array to register scheduling solution. Firstly, all columns are selected. So for each column inside the matrix we choose an element with the highest value and then mark the column of selected maximum element and the column number is registered in scheduling array. After all columns get processed, we will get scheduling solution from scheduling array and get the job completion time from scheduling solution.

If the number of jobs in proportion to the number of network nodes is small, we propose to assign job in a "First job-First process" manner. In a grid environment, a scheduler can have a different analysis criteria and decisions (access policies, access cost, required resource, processing speed, etc.). To formulate our algorithm, we propose to dynamically update job lists and network node list,  $J$  list and  $G$  list. All jobs and network nodes list are sorted in terms of job length, processing speed/access cost (based on multi-criteria analysis and decisions) in an ascending manner. The frequency of updating lists is highly depended on grid status, accessibility to grid network nodes and jobs.

Jlist1: maintains a list of all jobs which should be processed.

Jlist2: maintains a list of the jobs which have been assigned previously ( $Jlist3=Jlist1-Jlist2$ )

Glist1=maintains a list of network available nodes.

Glist2=maintains a list of the previously assigned node to jobs.

Glist3= a list of free node ( $Glist3=Glist1-Glist2$ )[5]

#### A. Scheduling using a fuzzy FPSO algorithm

In accordance to the represented scheduling schema for PSO in algorithm 1, we survey PSO and FPSO1.

Algorithm 1. A scheduling scheme based on fuzzy discrete PSO and FPSO1 [5]

0. If the grid is active and (jlist1=0) and no new jobs have been submitted, wait for new jobs to be submitted. Otherwise, update Glist1 and jlist1.
1. If (Glist1=0), wait until grid nodes are available. If jlist1>0, update jlist2. If jlist2< Glist1 allocate the jobs on a first – come –first served basis and if possible allocate the longest job on the fastest grid node according to the LJFN heuristic .

If jlist1> Glist1, job allocation is to be made by the following fuzzy discrete PSO algorithm detailed below .Take jobs and available grid nodes from jlist2 and Glist3. If  $m * n$  (m is the number of the grid nodes ,n is number of the jobs) is larger than the dimension threshold DT, the job and the grid nodes are grouped in to the fuzzy discrete PSO algorithm loop and the single node flow time is accumulated. The LFJN-SJFN heuristic is applied alternatively after a batch of job and nodes are allocated.

2. At t=0, represent the jobs and the nodes using fuzzy matrix.
3. Begin fuzzy discrete FPSO loop.
  - 3.0 Initialize the size of the particle swarm n and other parameters.
  - 3.1 Initialize a random position matrix and random velocity matrix for each particle, and then normalize matrices
  - 3.2 Repeat

3.2.0 T++;

3.2.1 Defuzzify the position, and calculate the make span and total flow time for each particle (the feasible solution)

3.2.2

$$X^* \operatorname{argmin}_{i=1}^n (f(X^*(t-1)), f(X_1(t)), f(X_2(t)) \dots f(X_i(t)) \dots f(X_n(t)));$$

3.2.3 For each particle

$$X_i^\#(t) = \operatorname{argmin}_{i=1}^n (f(X_i^\#(t-1)), f(X_i(t)))$$

3.2.4 For each particle, update each element in its position matrix and its velocity matrix according to equations;

3.2.5 Normalize the position matrix for each particle;

3.3 Until terminating criteria.

4. End of the fuzzy discrete PSO loop.
5. Check the feasibility and of the generated schedule with respect to grid node availability and user specified requirements. Then allocate the jobs to the grid nodes and update jlist2, jlist3, Glist2 and Glist3. Un-allocated jobs (infeasible schedules or grid node non-availability) shall be transferred to jlist1 for re-scheduling or dealt with separately.
6. Repeat steps 0-5 as long as the grid is active.

The parameters and values we considered for implementing algorithm 1 is listed on table 9.

TABLE 9. Parameter setting for GA genetic algorithm, A simulated annealing, PSO particle swarm optimization, FPSO1 hybrid particle swarm optimization.

Algorithm	Parameter	Parameter value
GA	population size	20
	Probability of crossover	0.8
	Probability of mutation	0.02
	Scale for mutations	0.01
SA	Operation number before temperature adjustment	20
	The number of cycles	10
	Temperature reduction factor	0.85
	Vector for control step of length adjustment	2
	Initial temperature	50
PSO	Swarm size	20
	Self-recognition coefficient c1	1.49
	Social coefficient c2	1.49
	Inertia weight w	0.1 < --- -0.9
FPSO1	Swarm size	20
	Self-recognition coefficient c1	1.49
	Social coefficient c2	1.49
	Inertia weight w	0.1 < --- -0.9

In each experiment, the algorithm was repeated for 10 times with random different grain. Each experiment is a constant number of  $50 * M * N$  (M is the number of

nodes of processors in grids and N is the number of jibs). The best range is obtained for 10 iteration along with the computed faults. The most important issue in

grid is that we should schedule jobs in such a way that the most rapid time is attained and also be executable. Thus, we have considered the total time of 10 runs as one of the performance optimization criteria.

To show this, we have started our experiments with 3 computing nodes and 13 jobs which is shown like (3, 13). The nodes speed CPUT: 4,3,2 and the length of 13 jobs in a cycle is 60,52,48,42,36,30,28,24,16,12,6 respectively.

Fig. 4 illustrated the functionality GA,SA,PSO,PFPSO1 Algorithms.

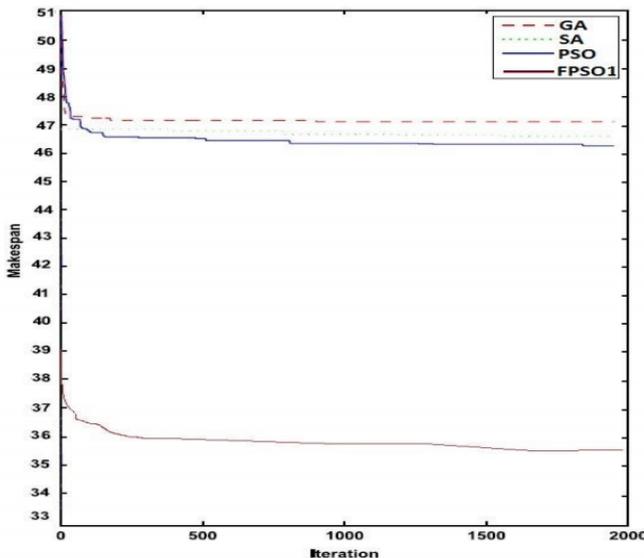


TABLE 10. An optimal scheduler for (3, 13)

Grid Node	Job												
	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13
G1	0	0	1	0	0	0	1	1	0	1	0	0	1
G2	1	0	0	1	1	0	0	0	1	0	1	0	0
G3	0	1	0	0	0	1	0	0	0	0	10	1	0

The method of scheduling is shown in Fig. 5 [5].

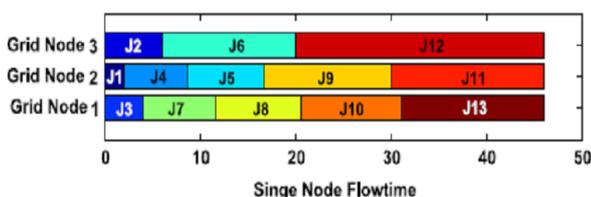


Fig. 5. The scheduling solution for (3, 13).

## VI. CONCLUSION

In this paper, a new fuzzy FPSO based approach was presented. According to the obtained results, it can be said that using fuzzy function can improve standard PSO algorithm behavior and make it possible for PSO algorithm react to abnormal situations. Also, as there are several parameters engaged in decision making, fuzzy logic FPSO1 is one of the best methods for non-linear functions in PSO. The method is

Fig. 4. Performance for GA, SA, PSO, FPSO1.

- The range of best results for GA algorithm for 10 runs is (49,47.3333,47,47,47,46,47.3333,47,46,47) with 47.1167 in average.
- The range of best results for SA algorithm for 10 runs is (47,47,47.333,47,46.6667,46,46,46,46.5,46.5) with 46.6 in average.
- The range of best results for PSO algorithm for 10 runs is (46.6667,46.5,46.5,46.5,46.5,46.5,46,46,46,46) with 46.2667 in average.
- The range of best results for GA algorithm for 10 runs is (36.3331,35.5,35,36.5,36,35,35.5,36,36,36) with 35.81661 in average.

The most optimal result in between above algorithm was 35 which was obtained by FPSO1 and after that is 46, while GA in 2 iteration, SA in 3 iteration and PSO in 5 iteration where succeed to get 46 as the best result.

Table 10. represent one of the best scheduling for (13, 3) in which, the job has been appointed to to related node [5].

recommended, that we produce an optimal scheduling in a dynamic manner, in such a way that all jobs will be accomplished in the less time as possible and also the available resource effectively get used. Empirical results show that the recommended method can be used for jobs scheduling.

## REFERENCES

[1] I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a New Computing Infrastructure, 2nd ed., Morgan Kaufmann, 2003.

[2] V.D. Martino, M. Mililotti, Sub optimal scheduling in a grid using genetic algorithms, Parallel Computing 30 (5\_6) (2004) 553\_565.

[3] Y. Gao, H.Q. Rong, J.Z. Huang, Adaptive Grid job scheduling with genetic algorithms, Future Generation Computer Systems 21 (1) (2005) 151\_161.

[4] A. Abraham, H. Liu, M. Zhao, Particle swarm scheduling for work-flow applications in distributed computing environments, in: *Metaheuristics for Scheduling: Industrial and Manufacturing Applications*, in: *Studies in Computational Intelligence*, Springer Verlag, Germany, 2008, pp. 327\_342.

[5] H. Liu, A. Abraham, A.E. Hassanien. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, 2010, pp.1336\_1343.

[6] T. Kiink, J. S. Vesterstroem, J. Riget, "Particle Swam Optimization with Spatial Particle Extension", *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1474-1479, 2002.

[7] L. Hongbo, M. Abraham, "Fuzzy Adaptive Turbulent Particle Swarm Optimization", *IEEE*, pp. 39-47, 2005.

[8] Q. Kang, L. Wang, Q. Wu, "Research on Fuzzy Adaptive Optimization Strategy of Particle Swarm Algorithm", *International Journal of Information Technology*, pp. 65-76, 2006.

[9] H. Liu, A. Abraham, W. Zhang, "A Fuzzy Adaptive Turbulent Particle Swarm Optimization", *Int. J. Innovative Computing and Applications*, pp.39-47, 2007.

[10] R. Brits, A. P. Engelbrecht, F. V. D. Bergh, "Locating Multiple Optima using particle Swarm Optimization", *Applied Mathematics and Computation* 189, Elsevier, pp. 1859-1883, 2007.

[11] M. Sugeno, "Industrial Applications of Fuzzy Control", Elsevier, New York, 1985.

[12] R. Eberhart, S. Yuhui, "Fuzzy Adaptive Particle Swarm Optimization", *IEEE*, pp. 101-106, 2001

[13] M. H. Noroozi Beyrami. M. R. Meybodi "Improving Particle Swarm Optimization using Fuzzy Logic" *Electrical and Computer Engineering and Information, Technology Department, Islamic Azad University Osku, Iran*, pp.40-52, 2008

[14] Hosein Nezamabadi, Majid Rostami Shahreabaki, "Generalization of the algorithm GCBPSO" *12 Computer Engineering Conference in Iran*, pp. 29-35, 2007

[15] X. Feng, J. Zhang, Z. Yang, "Adaptive Particle Swarm Optimization on Individual Level", *IEEE, China*, pp. 1215- 1218, 2002.

[16] F. Bergh, A. Engelbrecht, "A new locally convergent particle swarm optimizer", *IEEE*, 2002.

[17] D. Bratton, J. Kennedy, "Defining a Standard for Particle Swarm Optimization", *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, 2007.

[18] Z. Li, Y. Huan, H. Shang, "Optimal Choice of Parameters for Particle Swarm Optimization", *Journal of Zhejiang University SCIENCE ISSN 1009-3095*, pp. 528-534, 2005.

[19] W. Chung, R. Chang, A new mechanism for resource monitoring in Grid computing, *Future Generation Computer Systems* 25 (1) (2009) 1\_7.